In [44]:
```
# importing the required libaries
import os
import numpy as np
import pandas as pd
```

In [45]:
```
os.getcwd()
```

Out[45]:
```
'd:\\DATA_S\\Python\\Black Friday'
```

In [46]:
```
# read the CSV file, to load the data-set
df=pd.read_csv("blackfriday.csv")
```

In [47]:
```
# shape of the dataframe
df.shape
```

Out[47]:
```
(550068, 12)
```

In [48]:
```
# checking dimension
df.ndim
```

Out[48]:
```
2
```

In [49]:
```
#checking 1st 10 row
df.head(10)
```

Out[49]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
5	1000003	P00193542	M	26-35	15	A	
6	1000004	P00184942	M	46-50	7	B	
7	1000004	P00346142	M	46-50	7	B	
8	1000004	P0097242	M	46-50	7	B	
9	1000005	P00274942	M	26-35	20	A	

In [50]:
```
#checking the last 10 rows
df.tail(10)
```

Out[50]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
550058	1006024	P00372445	M	26-35	12	A	
550059	1006025	P00370853	F	26-35	1	B	
550060	1006026	P00371644	M	36-45	6	C	
550061	1006029	P00372445	F	26-35	1	C	
550062	1006032	P00372445	M	46-50	7	A	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

In [51]:
```
#checking for duplicates or total no. of duplicate rows
sum(df.duplicated())
```

Out[51]:
```
0
```

Creating a new Datasets with all product related columns

In [52]:
```
#checking the columns present in the dataframe
df.columns
```

Out[52]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase'],
      dtype='object')
```

In [53]:
```
#product related columns
[i for i in df.columns if 'Product'in i]
```

Out[53]:
```
['Product_ID',
 'Product_Category_1',
 'Product_Category_2',
 'Product_Category_3']
```

creating another dataset which will be a subset of the original and will have columns realted to products

In [54]:
```
#creating product related dataframe
df_product= df[['Product_ID','Product_Category_1','Product_Category_2','Product_
```

In [55]:
```
df_product
```

Out[55]:

	Product_ID	Product_Category_1	Product_Category_2	Product_Category_3
0	P00069042	3	NaN	NaN
1	P00248942	1	6.0	14.0
2	P00087842	12	NaN	NaN
3	P00085442	12	14.0	NaN
4	P00285442	8	NaN	NaN
...	...	...	...	...
550063	P00372445	20	NaN	NaN
550064	P00375436	20	NaN	NaN
550065	P00375436	20	NaN	NaN
550066	P00375436	20	NaN	NaN
550067	P00371644	20	NaN	NaN

550068 rows × 4 columns

In [56]:
```
#using info function to figure out missing or null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     550068 non-null  int64  
 1   Product_ID                  550068 non-null  object 
 2   Gender                      550068 non-null  object 
 3   Age                         550068 non-null  object 
 4   Occupation                  550068 non-null  int64  
 5   City_Category               550068 non-null  object 
 6   Stay_In_Current_City_Years  550068 non-null  object 
 7   Marital_Status              550068 non-null  int64  
 8   Product_Category_1          550068 non-null  int64  
 9   Product_Category_2          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    550068 non-null  int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

DESCRIPTIVE STATISTICS

In [57]:
```
df.describe()
```

Out[57]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Catego
count	5.500680e+05	550068.000000	550068.000000	550068.000000	376430.00
mean	1.003029e+06	8.076707	0.409653	5.404270	9.84
std	1.727592e+03	6.522660	0.491770	3.936211	5.08
min	1.000001e+06	0.000000	0.000000	1.000000	2.00
25%	1.001516e+06	2.000000	0.000000	1.000000	5.00
50%	1.003077e+06	7.000000	0.000000	5.000000	9.00
75%	1.004478e+06	14.000000	1.000000	8.000000	15.00
max	1.006040e+06	20.000000	1.000000	20.000000	18.00

In [58]:
```
# descriptive stat for object datatype
df.describe(include=[object])
```

Out[58]:

	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years
count	550068	550068	550068	550068	550068
unique	3631	2	7	3	5
top	P00265242	M	26-35	B	1
freq	1880	414259	219587	231173	193821

% distribution of each product ID available in dataset and find the highest occuring product_ID in the Dataset

In [59]:
```
# NUMBER OF TIMES A PRODUCT WAS PURCHASED
df['Product_ID'].value_counts()
```

Out[59]:
```
Product_ID
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: count, Length: 3631, dtype: int64
```

In [60]:
```
# converting the above in percentage
round(df['Product_ID'].value_counts(normalize=True)*100,3)
```

Out[60]:
```
Product_ID
P00265242    0.342
P00025442    0.294
P00110742    0.293
P00112142    0.284
P00057642    0.267
             ...  
P00314842    0.000
P00298842    0.000
P00231642    0.000
P00204442    0.000
P00066342    0.000
Name: proportion, Length: 3631, dtype: float64
```

product id with P0026524 has the highest percentage this implies its purchased more by the customers..

HANDLING MISSING VALUES

In [61]:
```
#count of null values
df.isnull().sum()
```

Out[61]:
```
User_ID                            0
Product_ID                         0
Gender                             0
Age                                0
Occupation                         0
City_Category                      0
Stay_In_Current_City_Years         0
Marital_Status                     0
Product_Category_1                 0
Product_Category_2            173638
Product_Category_3            383247
Purchase                           0
dtype: int64
```

In [62]:
```
#Analysisng the Product_Category_3 & 2 columns
 df.Product_Category_2
```

```
  Cell In[62], line 2
    df.Product_Category_2
    ^
IndentationError: unexpected indent
```

In [ ]:
```
df.Product_Category_3
```

Out[ ]:
```
0          NaN
1         14.0
2          NaN
3          NaN
4          NaN
          ... 
550063     NaN
550064     NaN
550065     NaN
550066     NaN
550067     NaN
Name: Product_Category_3, Length: 550068, dtype: float64
```

In [ ]:
```
#dropping Product_Category_2
df_temp= df.drop('Product_Category_2',axis=1,inplace=False)
```

In [ ]:
```
df_temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 11 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     550068 non-null  int64  
 1   Product_ID                  550068 non-null  object 
 2   Gender                      550068 non-null  object 
 3   Age                         550068 non-null  object 
 4   Occupation                  550068 non-null  int64  
 5   City_Category               550068 non-null  object 
 6   Stay_In_Current_City_Years  550068 non-null  object 
 7   Marital_Status              550068 non-null  int64  
 8   Product_Category_1          550068 non-null  int64  
 9   Product_Category_3          166821 non-null  float64
 10  Purchase                    550068 non-null  int64  
dtypes: float64(1), int64(5), object(5)
memory usage: 46.2+ MB
```

In [ ]:
```
#get index for all the rows of Product_Category_3 missing values
df_temp[df_temp.Product_Category_3.isnull()].index
```

Out[ ]:
```
Index([     0,      2,      3,      4,      5,      7,      8,      9,     10,
           11,
       ...
       550058, 550059, 550060, 550061, 550062, 550063, 550064, 550065, 550066,
       550067],
      dtype='int64', length=383247)
```

In [ ]:
```
df_temp= df_temp.drop(df_temp[df_temp.Product_Category_3.isnull()].index,axis=0,
```

In [ ]:
```
df_temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 166821 entries, 1 to 545914
Data columns (total 11 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     166821 non-null  int64  
 1   Product_ID                  166821 non-null  object 
 2   Gender                      166821 non-null  object 
 3   Age                         166821 non-null  object 
 4   Occupation                  166821 non-null  int64  
 5   City_Category               166821 non-null  object 
 6   Stay_In_Current_City_Years  166821 non-null  object 
 7   Marital_Status              166821 non-null  int64  
 8   Product_Category_1          166821 non-null  int64  
 9   Product_Category_3          166821 non-null  float64
 10  Purchase                    166821 non-null  int64  
dtypes: float64(1), int64(5), object(5)
memory usage: 15.3+ MB
```

Droped Product_Category_2 completely & Product_Category_3 we droped the missing values based on rows this entire process resulted in all the columns having equal values

Next , we try to Handle The missing values in Data frame df again using the imputting techiques such as mean/median/mode/ffill/bfill..

In [ ]:
```
#impute using forward filling
df= df.ffill()
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category_1            0
Product_Category_2            0
Product_Category_3            0
Purchase                      0
dtype: int64
```

In [ ]:
```
#impute using Backward filling
df= df.bfill()
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category_1            0
Product_Category_2            0
Product_Category_3            0
Purchase                      0
dtype: int64
```

Indexing and slicing using pandas

print age and occupation columnsusing LOC and select 1st, 5th & 10th rows with 1st, 4th and 7th columns using iloc

here we will see how to slice and dice the data and get the subset of the pandas dataframe

In [63]:
```
# make a copy of the dataframe
df1= df.copy()
```

In [64]:
```
df1
```

Out[64]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 12 columns

In [ ]:
```
# select 1st row of dataframe using LOC
df1.loc[0]
```

Out[ ]:
```
User_ID                         1000001
Product_ID                    P00069042
Gender                                F
Age                                0-17
Occupation                           10
City_Category                         A
Stay_In_Current_City_Years            2
Marital_Status                        0
Product_Category_1                    3
Product_Category_2                  NaN
Product_Category_3                  NaN
Purchase                           8370
Name: 0, dtype: object
```

In [ ]:
```
#1st and product ID
df1.loc[0,'Product_ID']
```

Out[ ]:
```
'P00069042'
```

In [ ]:
```
# printing purchase for all the rows
df1.loc[:,'Purchase']
```

Out[ ]:
```
0          8370
1         15200
2          1422
3          1057
4          7969
          ...  
550063      368
550064      371
550065      137
550066      365
550067      490
Name: Purchase, Length: 550068, dtype: int64
```

In [ ]:
```
# printing top 10 rows for purchase
df1.loc[:10,'Purchase']
```

Out[ ]:
```
0      8370
1     15200
2      1422
3      1057
4      7969
5     15227
6     19215
7     15854
8     15686
9      7871
10     5254
Name: Purchase, dtype: int64
```

In [ ]:
```
# print the age and occupation
df1.loc[:,['Age','Occupation']]
```

Out[ ]:

	Age	Occupation
0	0-17	10
1	0-17	10
2	0-17	10
3	0-17	10
4	55+	16
...	...	...
550063	51-55	13
550064	26-35	1
550065	26-35	15
550066	55+	1
550067	46-50	0

550068 rows × 2 columns

In [ ]:
```
# printing the 1st 5 rows of age and occupation
df1.loc[[0,1,2,3,4,5],['Age','Occupation']]
```

Out[ ]:

	Age	Occupation
0	0-17	10
1	0-17	10
2	0-17	10
3	0-17	10
4	55+	16
5	26-35	15

In [ ]:
```
# doing the same as above
df1.loc[:,['Age','Occupation']].head()
```

Out[ ]:

	Age	Occupation
0	0-17	10
1	0-17	10
2	0-17	10
3	0-17	10
4	55+	16

In [ ]:
```
# printing all the columns for 2nd ,3rd and 4th rows
df1.loc[2:4]
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	

In [ ]:
```
# printing the 1st row of the datafram using iloc
df1.iloc[0]
```

Out[ ]:
```
User_ID                          1000001
Product_ID                     P00069042
Gender                                 F
Age                                 0-17
Occupation                            10
City_Category                          A
Stay_In_Current_City_Years             2
Marital_Status                         0
Product_Category_1                     3
Product_Category_2                   NaN
Product_Category_3                   NaN
Purchase                            8370
Name: 0, dtype: object
```

In [ ]:
```
#the last of the dataframe
df1.iloc[-1]
```

Out[ ]:
```
User_ID                          1006039
Product_ID                     P00371644
Gender                                 F
Age                                46-50
Occupation                             0
City_Category                          B
Stay_In_Current_City_Years            4+
Marital_Status                         1
Product_Category_1                    20
Product_Category_2                   NaN
Product_Category_3                   NaN
Purchase                             490
Name: 550067, dtype: object
```

In [ ]:
```
# 1st 5columns of the dataframe with all rows using iloc
df1.iloc[:, :5]
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550068 rows × 5 columns

In [ ]:
```
#select 1st ,5th and 10th rows with 1st , 4th and 7th columns using lioc
df1.iloc[[0,4,9],[0,3,6]]
```

Out[ ]:

	User_ID	Age	Stay_In_Current_City_Years
0	1000001	0-17	2
4	1000002	55+	4+
9	1000005	26-35	1

Now we fetch row having maximum purchase amountwith completerow details

pandas provide two functions idmax() & idmin() that return index of 1st occurence of maximum or minimum values over requested axis. NULL values are ecluded from the out- put

In [ ]:
```
# index of 1st occurence of maximum value of purchase column
df1['Purchase'].idxmax()
```

Out[ ]:
```
87440
```

In [ ]:
```
# maximum value of the purchase column
df1.Purchase[df1['Purchase'].idxmax()]
```

Out[ ]:
```
23961
```

In [ ]:
```
#row with maximum purchase value
df1[df1['Purchase']==23961]
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
87440	1001474	P00052842	M	26-35	4	A	
93016	1002272	P00052842	M	26-35	0	C	
370891	1003160	P00052842	M	26-35	17	C	

GET PURCHASE AMOUNT FROM 3RD ROW

Pandas also provide at() and iat() function to access single value for a row and column pairby label or integer position

In [ ]:
```
# get value at the 3rd row and purchase column pair
df1.at[2,'Purchase']
```

Out[ ]:
```
1422
```

In [ ]:
```
df1.head()
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_'
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	

```
In [ ]: #3rd row and 11th column pair
        df1.at[2,'Product_Category_3']

Out[ ]: nan

In [ ]: # doing the same as above but using iat
        df1.iat[2,10]

Out[ ]: nan

In [ ]: # finding the purchase amount for user_id 1006039 and product _id P00371644
        df.loc[((df1['User_ID']==1006039) & (df1['Product_ID'] == 'P00371644')),'Purchas

Out[ ]: 550067    490
        Name: Purchase, dtype: int64

In [ ]: # checking for the above, 550067  is the index of 490 (purcchase amount)
        df1.loc[550067 :]

Out[ ]:
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
550067	1006039	P00371644	F	46-50	0	B	

FIND THE USERS IN CITY A WITH MORE THAN 4YRS AND PURCHASE AMOUNT MORE THAN 1000

```
In [ ]: # purchase amount wit given user_id & product_id
        df1[(df1['City_Category']== 'A') & (df1['Stay_In_Current_City_Years']=='4+') & (
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
98	1000022	P00351142	M	18-25	15	A	
100	1000022	P00195942	M	18-25	15	A	
102	1000022	P0098242	M	18-25	15	A	
103	1000022	P00262242	M	18-25	15	A	
416	1000073	P00351142	M	18-25	4	A	
...	...	...	...	...	...	...	
545791	1006019	P00279442	M	26-35	0	A	
545792	1006019	P00262342	M	26-35	0	A	
545793	1006019	P00028842	M	26-35	0	A	
545794	1006019	P00070342	M	26-35	0	A	
545832	1006028	P0097242	M	18-25	4	A	

6947 rows × 12 columns

discard all the users of city B with purchase less than 5000 and 3yrs time period

In [ ]:

```
#get the purchase with given user_id and the product_id
df1[(~(df1['Gender']=='F') & (df1['City_Category']=='B') & (df1['Stay_In_Current
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
112	1000023	P00278942	M	36-45	0	B	
113	1000023	P00004742	M	36-45	0	B	
114	1000023	P00198042	M	36-45	0	B	
115	1000023	P00177442	M	36-45	0	B	
116	1000023	P00000642	M	36-45	0	B	
...	...	...	...	...	...	...	
549883	1005776	P00371644	M	18-25	12	B	
549903	1005801	P00370853	M	26-35	16	B	
549909	1005811	P00375436	M	18-25	4	B	
549933	1005849	P00370293	M	36-45	17	B	
549945	1005868	P00370853	M	36-45	14	B	

31853 rows × 12 columns

FIND THE RECORD OF THE DATA SET BELEOW

[1006038,'P00375436','F','55+','1','C','2','0',20,2.0,11.0,365]

In [ ]:

```
# we will use isin() to slove the above
values = ['1006038','P00375436','F','55+','1','C','2','0',20,2.0,11.0,365]
df1_indexed= df1.isin(values)
df1_indexed
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
0	False	False	True	False	False	False	
1	False	False	True	False	False	False	
2	False	False	True	False	False	False	
3	False	False	True	False	False	False	
4	False	False	False	True	False	True	
...	...	...	...	...	...	...	
550063	False	False	False	False	False	False	
550064	False	True	True	False	False	True	
550065	False	True	True	False	False	False	
550066	False	True	True	True	False	True	
550067	False	False	True	False	False	False	

550068 rows × 12 columns

In [ ]:
```
# now we use all condition to find the columns for the given Data set
df1_indexed =df1.isin(values).all(axis=1)
df1[df1_indexed]
```

Out[ ]:

User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Ye

In [ ]:
```
# use mask function to get only the rows with occupation 10
new_df= df1.mask(df1['Occupation']!=10)
new_df.head(10)
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City
0	1000001.0	P00069042	F	0-17	10.0	A	
1	1000001.0	P00248942	F	0-17	10.0	A	
2	1000001.0	P00087842	F	0-17	10.0	A	
3	1000001.0	P00085442	F	0-17	10.0	A	
4	NaN	NaN	NaN	NaN	NaN	NaN	
5	NaN	NaN	NaN	NaN	NaN	NaN	
6	NaN	NaN	NaN	NaN	NaN	NaN	
7	NaN	NaN	NaN	NaN	NaN	NaN	
8	NaN	NaN	NaN	NaN	NaN	NaN	
9	NaN	NaN	NaN	NaN	NaN	NaN	

In [ ]:

```
# sort the dataset row wise
# via default sort is row-wise
df1.sort_index()
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 12 columns

In [ ]:
```
# sort via column
df1.sort_index(axis=1)
```

Out[ ]:

	Age	City_Category	Gender	Marital_Status	Occupation	Product_Category_1
0	0-17	A	F	0	10	3
1	0-17	A	F	0	10	1
2	0-17	A	F	0	10	12
3	0-17	A	F	0	10	12
4	55+	C	M	0	16	8
...	...	...	...	...	...	...
550063	51-55	B	M	1	13	20
550064	26-35	C	F	0	1	20
550065	26-35	B	F	1	15	20
550066	55+	C	F	0	1	20
550067	46-50	B	F	1	0	20

550068 rows × 12 columns

In [ ]:

```
# sort row by decending order
df1.sort_index(ascending=False)
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
550067	1006039	P00371644	F	46-50	0	B	
550066	1006038	P00375436	F	55+	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550064	1006035	P00375436	F	26-35	1	C	
550063	1006033	P00372445	M	51-55	13	B	
...	...	...	...	...	...	...	
4	1000002	P00285442	M	55+	16	C	
3	1000001	P00085442	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
0	1000001	P00069042	F	0-17	10	A	

550068 rows × 12 columns

FINDING TOP 20 MOSTREVENUE GENERATED CUSTOMER AND THEIR PURCHASED PRODUCT_ID

In [ ]:
```
#sorting the dataset using purchase column
df1.sort_values(by='Purchase')
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
549221	1004806	P00370293	M	26-35	17	C	
549477	1005184	P00370293	M	18-25	20	B	
547819	1002802	P00370853	M	36-45	20	B	
548027	1003105	P00370853	M	36-45	12	C	
547538	1002402	P00370853	M	46-50	17	B	
...	...	...	...	...	...	...	
292083	1003045	P00052842	M	46-50	1	B	
503697	1005596	P00117642	M	36-45	12	B	
370891	1003160	P00052842	M	26-35	17	C	
87440	1001474	P00052842	M	26-35	4	A	
93016	1002272	P00052842	M	26-35	0	C	

550068 rows × 12 columns

In [ ]:

```
# sort by purchse and age
df1.sort_values(by=['Purchase','Age']).head(5)
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
546045	1000194	P00370853	F	0-17	10	C	
546449	1000775	P00370853	M	0-17	17	C	
550024	1005973	P00370293	M	0-17	10	C	
546301	1000561	P00370853	F	18-25	14	C	
546333	1000608	P00370293	M	18-25	4	C	

In [ ]:

```
# sorting desending order
df1.sort_values(by='Purchase',ascending= False)
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
370891	1003160	P00052842	M	26-35	17	C	
93016	1002272	P00052842	M	26-35	0	C	
87440	1001474	P00052842	M	26-35	4	A	
503697	1005596	P00117642	M	36-45	12	B	
321782	1001577	P00052842	M	55+	0	C	
...	...	...	...	...	...	...	
546379	1000671	P00370853	M	18-25	4	C	
546185	1000391	P00370293	M	46-50	11	C	
547032	1001649	P00370293	M	18-25	19	C	
546181	1000387	P00370293	F	36-45	7	C	
549221	1004806	P00370293	M	26-35	17	C	

550068 rows × 12 columns

In [ ]:

```python
#3 top 20 using iloc
top20= df1.sort_values(by='Purchase', ascending=False).iloc[:20,:]
```

In [ ]:

```python
top20
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
370891	1003160	P00052842	M	26-35	17	C	
93016	1002272	P00052842	M	26-35	0	C	
87440	1001474	P00052842	M	26-35	4	A	
503697	1005596	P00117642	M	36-45	12	B	
321782	1001577	P00052842	M	55+	0	C	
349658	1005848	P00119342	M	51-55	20	A	
292083	1003045	P00052842	M	46-50	1	B	
298378	1003947	P00116142	M	26-35	0	C	
437804	1001387	P00086242	F	51-55	13	B	
229329	1005367	P00085342	M	18-25	4	A	
416883	1004117	P00161842	M	18-25	4	B	
7542	1001178	P00116142	M	55+	0	C	
373300	1003511	P00085342	M	51-55	0	C	
33268	1005102	P00052842	M	26-35	12	C	
388010	1005716	P00052842	M	0-17	10	C	
449656	1003301	P00086242	F	26-35	2	B	
366333	1002359	P00085342	M	55+	13	C	
54364	1002274	P00052842	M	18-25	2	B	
56879	1002788	P00085342	M	55+	1	B	
68926	1004520	P00116142	M	26-35	4	C	

In [ ]:

```python
# get user _id for this top 20
top20.User_ID.values
```

```
Out[ ]: array([1003160, 1002272, 1001474, 1005596, 1001577, 1005848, 1003045,
               1003947, 1001387, 1005367, 1004117, 1001178, 1003511, 1005102,
               1005716, 1003301, 1002359, 1002274, 1002788, 1004520], dtype=int64)
```

```
In [ ]: # top 20 products
        top20.Product_ID.value_counts()
```

```
Out[ ]: Product_ID
        P00052842    8
        P00085342    4
        P00116142    3
        P00086242    2
        P00117642    1
        P00119342    1
        P00161842    1
        Name: count, dtype: int64
```

FIND WHICH AGE GROUP IS MORE ACTIVE IN PURCHASING PRODUCT FROM THE WEBSITE

```
In [ ]: #USING UNIQUE TO GET DISTINCT VALUES
        df1['Gender'].unique()
```

```
Out[ ]: array(['F', 'M'], dtype=object)
```

```
In [ ]: # count of unique values of the above gender class
        df1['Gender'].value_counts()
```

```
Out[ ]: Gender
        M    414259
        F    135809
        Name: count, dtype: int64
```

```
In [ ]: # sort w.r.t count
        df1["Gender"].value_counts(ascending=True)
```

```
Out[ ]: Gender
        F    135809
        M    414259
        Name: count, dtype: int64
```

```
In [ ]: #age count sorting in ascending order
        df1['Age'].value_counts(ascending=True)
```

```
Out[ ]: Age
        0-17      15102
        55+       21504
        51-55     38501
        46-50     45701
        18-25     99660
        36-45    110013
        26-35    219587
        Name: count, dtype: int64
```

```
In [ ]: # age count sorting in decending order
        df1['Age'].value_counts(ascending=False)
```

Out[ ]:
```
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
```

In [ ]:
```
# we replace F for FEMALE and M to be MALE FOR Gender column
df1['Gender']=df1['Gender'].replace('F','Female')
df1['Gender']=df1['Gender'].replace('M','Male')
```

In [ ]:
```
df1.head(20)
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City
0	1000001	P00069042	Female	0-17	10	A	
1	1000001	P00248942	Female	0-17	10	A	
2	1000001	P00087842	Female	0-17	10	A	
3	1000001	P00085442	Female	0-17	10	A	
4	1000002	P00285442	Male	55+	16	C	
5	1000003	P00193542	Male	26-35	15	A	
6	1000004	P00184942	Male	46-50	7	B	
7	1000004	P00346142	Male	46-50	7	B	
8	1000004	P0097242	Male	46-50	7	B	
9	1000005	P00274942	Male	26-35	20	A	
10	1000005	P00251242	Male	26-35	20	A	
11	1000005	P00014542	Male	26-35	20	A	
12	1000005	P00031342	Male	26-35	20	A	
13	1000005	P00145042	Male	26-35	20	A	
14	1000006	P00231342	Female	51-55	9	A	
15	1000006	P00190242	Female	51-55	9	A	
16	1000006	P0096642	Female	51-55	9	A	
17	1000006	P00058442	Female	51-55	9	A	
18	1000007	P00036842	Male	36-45	1	B	
19	1000008	P00249542	Male	26-35	12	C	

GENERATE THE LIST OF USER_IDWITH CORRESPONDING AGE ANDFIND THE TOTAL COUNT OF PURCHASETHAT THEY HAVE DONE

In [ ]:
```
# 1ST GET THE LIST OF USER_ID AND AGE USING TOLIST
df[['User_ID','Age']].values.tolist()
```

```
Out[ ]:  [[1000001, '0-17'],
          [1000001, '0-17'],
          [1000001, '0-17'],
          [1000001, '0-17'],
          [1000002, '55+'],
          [1000003, '26-35'],
          [1000004, '46-50'],
          [1000004, '46-50'],
          [1000004, '46-50'],
          [1000005, '26-35'],
          [1000005, '26-35'],
          [1000005, '26-35'],
          [1000005, '26-35'],
          [1000005, '26-35'],
          [1000006, '51-55'],
          [1000006, '51-55'],
          [1000006, '51-55'],
          [1000006, '51-55'],
          [1000007, '36-45'],
          [1000008, '26-35'],
          [1000008, '26-35'],
          [1000008, '26-35'],
          [1000008, '26-35'],
          [1000008, '26-35'],
          [1000008, '26-35'],
          [1000009, '26-35'],
          [1000009, '26-35'],
          [1000009, '26-35'],
          [1000009, '26-35'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000010, '36-45'],
          [1000011, '26-35'],
          [1000011, '26-35'],
          [1000011, '26-35'],
          [1000012, '26-35'],
          [1000012, '26-35'],
          [1000013, '46-50'],
          [1000013, '46-50'],
          [1000013, '46-50'],
          [1000014, '36-45'],
          [1000015, '26-35'],
          [1000015, '26-35'],
          [1000015, '26-35'],
          [1000015, '26-35'],
          [1000015, '26-35'],
          [1000015, '26-35'],
          [1000015, '26-35'],
          [1000015, '26-35'],
          [1000015, '26-35'],
          [1000016, '36-45'],
          [1000016, '36-45'],
          [1000017, '51-55'],
          [1000017, '51-55'],
          [1000017, '51-55'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000018, '18-25'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000019, '0-17'],
          [1000021, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000022, '18-25'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000023, '36-45'],
          [1000024, '26-35'],
          [1000024, '26-35'],
          [1000024, '26-35'],
          [1000025, '18-25'],
          [1000025, '18-25'],
          [1000025, '18-25'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000026, '26-35'],
          [1000027, '26-35'],
          [1000027, '26-35'],
          [1000027, '26-35'],
          [1000027, '26-35'],
          [1000027, '26-35'],
          [1000028, '26-35'],
          [1000028, '26-35'],
          [1000028, '26-35'],
          [1000028, '26-35'],
          [1000028, '26-35'],
          [1000029, '36-45'],
          [1000029, '36-45'],
          [1000029, '36-45'],
          [1000029, '36-45'],
          [1000030, '36-45'],
          [1000030, '36-45'],
          [1000030, '36-45'],
          [1000031, '55+'],
          [1000031, '55+'],
          [1000031, '55+'],
          [1000031, '55+'],
          [1000032, '26-35'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000033, '46-50'],
          [1000034, '18-25'],
          [1000034, '18-25'],
          [1000034, '18-25'],
          [1000034, '18-25'],
          [1000034, '18-25'],
          [1000034, '18-25'],
          [1000034, '18-25'],
          [1000034, '18-25'],
          [1000034, '18-25'],
          [1000034, '18-25'],
          [1000035, '46-50'],
          [1000035, '46-50'],
          [1000035, '46-50'],
          [1000035, '46-50'],
          [1000035, '46-50'],
          [1000035, '46-50'],
          [1000035, '46-50'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000036, '26-35'],
          [1000037, '26-35'],
          [1000039, '18-25'],
          [1000039, '18-25'],
          [1000041, '18-25'],
          [1000041, '18-25'],
          [1000042, '26-35'],
          [1000042, '26-35'],
          [1000042, '26-35'],
          [1000042, '26-35'],
          [1000042, '26-35'],
          [1000042, '26-35'],
          [1000042, '26-35'],
          [1000042, '26-35'],
          [1000042, '26-35'],
          [1000042, '26-35'],
          [1000043, '26-35'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000044, '46-50'],
          [1000045, '46-50'],
          [1000045, '46-50'],
          [1000045, '46-50'],
          [1000045, '46-50'],
          [1000045, '46-50'],
          [1000045, '46-50'],
          [1000045, '46-50'],
          [1000045, '46-50'],
          [1000045, '46-50'],
          [1000045, '46-50'],
          [1000046, '18-25'],
          [1000046, '18-25'],
          [1000046, '18-25'],
          [1000046, '18-25'],
          [1000047, '18-25'],
          [1000047, '18-25'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000048, '26-35'],
          [1000049, '18-25'],
          [1000049, '18-25'],
          [1000049, '18-25'],
          [1000049, '18-25'],
          [1000049, '18-25'],
          [1000049, '18-25'],
          [1000050, '26-35'],
          [1000050, '26-35'],
          [1000051, '0-17'],
          [1000052, '18-25'],
          [1000052, '18-25'],
          [1000052, '18-25'],
          [1000052, '18-25'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000053, '26-35'],
          [1000054, '51-55'],
          [1000054, '51-55'],
          [1000056, '36-45'],
          [1000056, '36-45'],
          [1000056, '36-45'],
          [1000056, '36-45'],
          [1000057, '18-25'],
          [1000057, '18-25'],
          [1000057, '18-25'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000058, '26-35'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000059, '51-55'],
          [1000060, '51-55'],
          [1000060, '51-55'],
          [1000061, '26-35'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000062, '36-45'],
          [1000063, '18-25'],
          [1000063, '18-25'],
          [1000063, '18-25'],
          [1000063, '18-25'],
          [1000064, '18-25'],
          [1000065, '36-45'],
          [1000065, '36-45'],
          [1000065, '36-45'],
          [1000065, '36-45'],
          [1000065, '36-45'],
          [1000065, '36-45'],
          [1000066, '26-35'],
          [1000067, '51-55'],
          [1000067, '51-55'],
          [1000067, '51-55'],
          [1000068, '18-25'],
          [1000068, '18-25'],
          [1000068, '18-25'],
          [1000069, '26-35'],
          [1000069, '26-35'],
          [1000070, '18-25'],
          [1000070, '18-25'],
          [1000071, '26-35'],
          [1000071, '26-35'],
          [1000071, '26-35'],
          [1000071, '26-35'],
          [1000072, '46-50'],
          [1000072, '46-50'],
          [1000073, '18-25'],
          [1000073, '18-25'],
          [1000073, '18-25'],
          [1000073, '18-25'],
          [1000073, '18-25'],
          [1000074, '36-45'],
          [1000074, '36-45'],
          [1000075, '0-17'],
          [1000075, '0-17'],
          [1000075, '0-17'],
          [1000075, '0-17'],
          [1000075, '0-17'],
          [1000075, '0-17'],
          [1000075, '0-17'],
          [1000075, '0-17'],
          [1000075, '0-17'],
          [1000075, '0-17'],
          [1000076, '36-45'],
          [1000076, '36-45'],
          [1000076, '36-45'],
          [1000076, '36-45'],
          [1000076, '36-45'],
          [1000077, '18-25'],
          [1000077, '18-25'],
          [1000077, '18-25'],
          [1000078, '46-50'],
          [1000078, '46-50'],
          [1000078, '46-50'],
          [1000078, '46-50'],
          [1000078, '46-50'],
          [1000078, '46-50'],
          [1000078, '46-50'],
          [1000078, '46-50'],
          [1000079, '46-50'],
          [1000079, '46-50'],
          [1000080, '55+'],
          [1000080, '55+'],
          [1000080, '55+'],
          [1000081, '26-35'],
          [1000082, '26-35'],
          [1000082, '26-35'],
          [1000083, '26-35'],
          [1000083, '26-35'],
          [1000083, '26-35'],
          [1000083, '26-35'],
          [1000083, '26-35'],
          [1000084, '18-25'],
          [1000084, '18-25'],
          [1000084, '18-25'],
          [1000084, '18-25'],
          [1000085, '18-25'],
          [1000086, '0-17'],
          [1000086, '0-17'],
          [1000087, '26-35'],
          [1000087, '26-35'],
          [1000087, '26-35'],
          [1000088, '46-50'],
          [1000088, '46-50'],
          [1000088, '46-50'],
          [1000089, '55+'],
          [1000089, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000090, '55+'],
          [1000091, '36-45'],
          [1000091, '36-45'],
          [1000091, '36-45'],
          [1000091, '36-45'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000092, '18-25'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000093, '26-35'],
          [1000094, '26-35'],
          [1000095, '46-50'],
          [1000095, '46-50'],
          [1000096, '26-35'],
          [1000096, '26-35'],
          [1000097, '36-45'],
          [1000097, '36-45'],
          [1000097, '36-45'],
          [1000097, '36-45'],
          [1000097, '36-45'],
          [1000097, '36-45'],
          [1000097, '36-45'],
          [1000097, '36-45'],
          [1000099, '0-17'],
          [1000099, '0-17'],
          [1000099, '0-17'],
          [1000099, '0-17'],
          [1000099, '0-17'],
          [1000100, '36-45'],
          [1000100, '36-45'],
          [1000100, '36-45'],
          [1000101, '18-25'],
          [1000101, '18-25'],
          [1000101, '18-25'],
          [1000101, '18-25'],
          [1000102, '36-45'],
          [1000102, '36-45'],
          [1000102, '36-45'],
          [1000102, '36-45'],
          [1000102, '36-45'],
          [1000102, '36-45'],
          [1000102, '36-45'],
          [1000103, '46-50'],
          [1000103, '46-50'],
          [1000103, '46-50'],
          [1000103, '46-50'],
          [1000105, '46-50'],
          [1000105, '46-50'],
          [1000105, '46-50'],
          [1000105, '46-50'],
          [1000106, '36-45'],
          [1000106, '36-45'],
          [1000106, '36-45'],
          [1000107, '46-50'],
          [1000107, '46-50'],
          [1000107, '46-50'],
          [1000107, '46-50'],
          [1000107, '46-50'],
          [1000107, '46-50'],
          [1000108, '26-35'],
          [1000109, '46-50'],
          [1000109, '46-50'],
          [1000109, '46-50'],
          [1000110, '26-35'],
          [1000111, '36-45'],
          [1000111, '36-45'],
          [1000112, '26-35'],
          [1000113, '18-25'],
          [1000113, '18-25'],
          [1000114, '26-35'],
          [1000114, '26-35'],
          [1000116, '26-35'],
          [1000116, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000117, '26-35'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000118, '36-45'],
          [1000119, '0-17'],
          [1000119, '0-17'],
          [1000119, '0-17'],
          [1000119, '0-17'],
          [1000119, '0-17'],
          [1000120, '26-35'],
          [1000121, '36-45'],
          [1000121, '36-45'],
          [1000122, '18-25'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000123, '36-45'],
          [1000124, '55+'],
          [1000125, '46-50'],
          [1000125, '46-50'],
          [1000125, '46-50'],
          [1000125, '46-50'],
          [1000125, '46-50'],
          [1000125, '46-50'],
          [1000126, '18-25'],
          [1000126, '18-25'],
          [1000127, '46-50'],
          [1000127, '46-50'],
          [1000127, '46-50'],
          [1000127, '46-50'],
          [1000127, '46-50'],
          [1000128, '55+'],
          [1000129, '26-35'],
          [1000129, '26-35'],
          [1000129, '26-35'],
          [1000129, '26-35'],
          [1000129, '26-35'],
          [1000130, '36-45'],
          [1000130, '36-45'],
          [1000130, '36-45'],
          [1000130, '36-45'],
          [1000130, '36-45'],
          [1000130, '36-45'],
          [1000130, '36-45'],
          [1000130, '36-45'],
          [1000130, '36-45'],
          [1000130, '36-45'],
          [1000130, '36-45'],
          [1000131, '18-25'],
          [1000131, '18-25'],
          [1000131, '18-25'],
          [1000131, '18-25'],
          [1000131, '18-25'],
          [1000131, '18-25'],
          [1000131, '18-25'],
          [1000131, '18-25'],
          [1000131, '18-25'],
          [1000131, '18-25'],
          [1000132, '26-35'],
          [1000132, '26-35'],
          [1000132, '26-35'],
          [1000132, '26-35'],
          [1000132, '26-35'],
          [1000133, '26-35'],
          [1000133, '26-35'],
          [1000133, '26-35'],
          [1000133, '26-35'],
          [1000133, '26-35'],
          [1000134, '26-35'],
          [1000134, '26-35'],
          [1000134, '26-35'],
          [1000134, '26-35'],
          [1000135, '18-25'],
          [1000135, '18-25'],
          [1000135, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000136, '18-25'],
          [1000137, '46-50'],
          [1000137, '46-50'],
          [1000137, '46-50'],
          [1000137, '46-50'],
          [1000137, '46-50'],
          [1000137, '46-50'],
          [1000137, '46-50'],
          [1000137, '46-50'],
          [1000137, '46-50'],
          [1000138, '18-25'],
          [1000138, '18-25'],
          [1000138, '18-25'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000139, '26-35'],
          [1000140, '36-45'],
          [1000140, '36-45'],
          [1000142, '26-35'],
          [1000142, '26-35'],
          [1000142, '26-35'],
          [1000142, '26-35'],
          [1000142, '26-35'],
          [1000143, '18-25'],
          [1000143, '18-25'],
          [1000143, '18-25'],
          [1000145, '18-25'],
          [1000145, '18-25'],
          [1000145, '18-25'],
          [1000145, '18-25'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000146, '36-45'],
          [1000147, '18-25'],
          [1000147, '18-25'],
          [1000147, '18-25'],
          [1000147, '18-25'],
          [1000147, '18-25'],
          [1000147, '18-25'],
          [1000147, '18-25'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000148, '51-55'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000149, '26-35'],
          [1000150, '36-45'],
          [1000150, '36-45'],
          [1000150, '36-45'],
          [1000150, '36-45'],
          [1000150, '36-45'],
          [1000150, '36-45'],
          [1000150, '36-45'],
          [1000150, '36-45'],
          [1000150, '36-45'],
          [1000150, '36-45'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000151, '26-35'],
          [1000152, '18-25'],
          [1000153, '0-17'],
          [1000153, '0-17'],
          [1000153, '0-17'],
          [1000154, '51-55'],
          [1000154, '51-55'],
          [1000154, '51-55'],
          [1000154, '51-55'],
          [1000155, '36-45'],
          [1000155, '36-45'],
          [1000155, '36-45'],
          [1000155, '36-45'],
          [1000155, '36-45'],
          [1000155, '36-45'],
          [1000155, '36-45'],
          [1000156, '46-50'],
          [1000156, '46-50'],
          [1000156, '46-50'],
          [1000157, '36-45'],
          [1000157, '36-45'],
          [1000157, '36-45'],
          [1000157, '36-45'],
          [1000157, '36-45'],
          [1000157, '36-45'],
          [1000157, '36-45'],
          [1000157, '36-45'],
          [1000157, '36-45'],
          [1000157, '36-45'],
          [1000157, '36-45'],
          [1000158, '55+'],
          [1000158, '55+'],
          [1000159, '46-50'],
          [1000160, '36-45'],
          [1000160, '36-45'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000161, '46-50'],
          [1000162, '18-25'],
          [1000162, '18-25'],
          [1000162, '18-25'],
          [1000162, '18-25'],
          [1000162, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000163, '18-25'],
          [1000165, '18-25'],
          [1000165, '18-25'],
          [1000165, '18-25'],
          [1000165, '18-25'],
          [1000165, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000166, '18-25'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000169, '26-35'],
          [1000170, '26-35'],
          [1000170, '26-35'],
          [1000171, '51-55'],
          [1000172, '26-35'],
          [1000173, '26-35'],
          ...]
```

In [ ]:
```
# count of purchase for all distinct user_id and age combination
df1[['User_ID','Age']].value_counts()
```

Out[ ]:
```
User_ID  Age  
1001680  26-35    1026
1004277  36-45     979
1001941  36-45     898
1001181  36-45     862
1000889  46-50     823
                  ... 
1002111  55+         7
1005391  26-35       7
1002690  26-35       7
1005608  18-25       7
1000708  26-35       6
Name: count, Length: 5891, dtype: int64
```

FINDING DIFFERENT STATISTICAL VALUES OF THE PURCHASE COLUMN

AGGREGATION IN PANDAS

```
In [69]: import numpy as np
```

```
In [68]: df1['Purchase'].describe()
```

```
Out[68]: count    550068.000000
         mean       9263.968713
         std        5023.065394
         min          12.000000
         25%        5823.000000
         50%        8047.000000
         75%       12054.000000
         max       23961.000000
         Name: Purchase, dtype: float64
```

TOTAL AMOUNT GENERATED VIA WEBSITE BY SELLING PRODUCT

```
In [ ]: #use np.sum aggregation to get total purchased amount
        df1['Purchase'].aggregate(np.sum)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_57636\2912300007.py:2: FutureWarning:
The provided callable <function sum at 0x000001CE264CEAC0> is currently using Ser
ies.sum. In a future version of pandas, the provided callable will be used direct
ly. To keep current behavior pass the string "sum" instead.
  df1['Purchase'].aggregate(np.sum)
```

```
Out[ ]: 5095812742
```

```
In [ ]:  # find sum and mean value  after doing aggregation over purchase column
         df1['Purchase'].aggregate([np.sum,np.mean])
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_57636\702596433.py:2: FutureWarning: T
he provided callable <function sum at 0x000001CE264CEAC0> is currently using Seri
es.sum. In a future version of pandas, the provided callable will be used directl
y. To keep current behavior pass the string "sum" instead.
  df1['Purchase'].aggregate([np.sum,np.mean])
C:\Users\User\AppData\Local\Temp\ipykernel_57636\702596433.py:2: FutureWarning: T
he provided callable <function mean at 0x000001CE264CFBA0> is currently using Ser
ies.mean. In a future version of pandas, the provided callable will be used direc
tly. To keep current behavior pass the string "mean" instead.
  df1['Purchase'].aggregate([np.sum,np.mean])
```

```
Out[ ]: sum     5.095813e+09
        mean    9.263969e+03
        Name: Purchase, dtype: float64
```

```
In [ ]: #Find mean value for Product_Category_1 ,Product_Category_2,Product_Category_3
        df1[['Product_Category_1','Product_Category_2','Product_Category_3']].aggregate(
C:\Users\User\AppData\Local\Temp\ipykernel_57636\3948570574.py:2: FutureWarning:
The provided callable <function mean at 0x000001CE264CFBA0> is currently using Se
ries.mean. In a future version of pandas, the provided callable will be used dire
ctly. To keep current behavior pass the string "mean" instead.
  df1[['Product_Category_1','Product_Category_2','Product_Category_3']].aggregate
([np.mean,np.sum])
C:\Users\User\AppData\Local\Temp\ipykernel_57636\3948570574.py:2: FutureWarning:
The provided callable <function sum at 0x000001CE264CEAC0> is currently using Ser
ies.sum. In a future version of pandas, the provided callable will be used direct
ly. To keep current behavior pass the string "sum" instead.
  df1[['Product_Category_1','Product_Category_2','Product_Category_3']].aggregate
([np.mean,np.sum])
```

Out[ ]:

	Product_Category_1	Product_Category_2	Product_Category_3
mean	5.404270e+00	9.842329e+00	1.266824e+01
sum	2.972716e+06	3.704948e+06	2.113329e+06

TAG RECORDS 'HIGH FOCUSED' TRANSACTION WHERE PURCHASE AMOUNT HAS BEEN MORE THAN 5000 . REMAINNING CAN BE TAGGED AS ' GENERAL TRANSACTION'

In [ ]:

```
#using apply function to Product_Category_1
df1.Product_Category_1.apply(lambda x:x*10)
```

Out[ ]:

```
0          30
1          10
2         120
3         120
4          80
         ... 
550063    200
550064    200
550065    200
550066    200
550067    200
Name: Product_Category_1, Length: 550068, dtype: int64
```

In [ ]:

```
df1.head(10)
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_
0	1000001	P00069042	Female	0-17	10	A	
1	1000001	P00248942	Female	0-17	10	A	
2	1000001	P00087842	Female	0-17	10	A	
3	1000001	P00085442	Female	0-17	10	A	
4	1000002	P00285442	Male	55+	16	C	
5	1000003	P00193542	Male	26-35	15	A	
6	1000004	P00184942	Male	46-50	7	B	
7	1000004	P00346142	Male	46-50	7	B	
8	1000004	P0097242	Male	46-50	7	B	
9	1000005	P00274942	Male	26-35	20	A	

In [ ]:
```
# adding new column 'category' which would help in adding tag for purchase amoun
df1['Category']=df1.Purchase.apply(lambda x :'HIGH FOCUSE' if x>5000 else 'GENER
df1.head()
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_
0	1000001	P00069042	Female	0-17	10	A	
1	1000001	P00248942	Female	0-17	10	A	
2	1000001	P00087842	Female	0-17	10	A	
3	1000001	P00085442	Female	0-17	10	A	
4	1000002	P00285442	Male	55+	16	C	

In [ ]:
```
# lets check value for high focused row
df1.Category.value_counts()
```

Out[ ]:
```
Category
HIGH FOCUSE            455145
GENERAL TRANSACTION     94923
Name: count, dtype: int64
```

BASED ON GENDER, CHECK THE TOTAL PURCHASE AMOUNT AND AVERAGE PURCHASING AMOUNT

In [67]:
```
# use group by on gender column
df1.groupby('Gender')
```

Out[67]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001B12AA5D3D0>
```

In [66]:
```
df1.groupby('Gender').groups
```

Out[66]:
```
{'F': [0, 1, 2, 3, 14, 15, 16, 17, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 65, 66, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 124, 125, 126, 147, 148, 149, 150, 151, 156, 157, 1
58, 163, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 219, 222, 223, 248,
249, 250, 251, 252, 253, 254, 255, 256, 257, 297, 298, 299, 355, 356, 357, 358,
359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 373, ...], 'M': [4, 5,
6, 7, 8, 9, 10, 11, 12, 13, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 50, 51,
52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 67, 68, 69, 85, 86, 87, 88,
89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106,
107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122,
123, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141,
142, 143, 144, 145, 146, 152, 153, ...]}
```

In [65]:
```
#get groups based on gender and age combination
df1.groupby(['Gender','Age']).groups
```

Out[65]:
```
{('F', '0-17'): [0, 1, 2, 3, 299, 423, 424, 425, 426, 427, 428, 429, 430, 431,
432, 467, 468, 539, 540, 541, 542, 543, 617, 618, 619, 620, 621, 1150, 1151, 13
04, 1305, 1306, 2905, 2907, 3010, 3715, 3804, 3805, 3806, 3807, 3808, 3835, 383
6, 4551, 4552, 4553, 4554, 4555, 5453, 6431, 6759, 6760, 6761, 6762, 6763, 676
4, 6765, 6766, 6767, 6768, 6769, 6770, 6771, 6772, 6773, 6774, 6775, 6776, 677
7, 6778, 6779, 6780, 6781, 6782, 6783, 6784, 6785, 6786, 6787, 6788, 6789, 679
0, 6791, 6792, 6793, 6794, 6795, 6796, 6797, 6798, 6799, 6800, 6801, 6802, 680
3, 6804, 6805, 6806, 6807, 6808, ...], ('F', '18-25'): [70, 71, 72, 73, 74, 75,
76, 77, 78, 79, 80, 81, 82, 83, 84, 179, 180, 181, 182, 183, 184, 185, 186, 18
7, 188, 222, 223, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 5
07, 508, 509, 547, 548, 549, 550, 625, 910, 911, 912, 913, 914, 1046, 1228, 126
7, 1268, 1269, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 150
0, 1552, 1553, 1554, 1555, 1556, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 167
2, 1673, 1674, 1675, 1676, 1677, 1678, 1822, 1903, 1904, 1905, 1947, 1948, 194
9, 1950, 1951, 1952, 1953, 1954, 1959, ...], ('F', '26-35'): [47, 48, 49, 124,
125, 126, 147, 148, 149, 150, 151, 163, 219, 297, 298, 406, 407, 454, 457, 458,
459, 460, 461, 529, 530, 585, 586, 691, 692, 693, 694, 695, 729, 730, 731, 732,
733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 840, 841, 842, 843, 844, 845,
846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861,
862, 863, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 104
4, 1045, 1085, 1086, 1087, 1088, 1364, 1365, 1369, 1565, 1627, 1628, 1629, 163
0, 1631, 1632, 1633, 1634, 1635, ...], ('F', '36-45'): [29, 30, 31, 32, 33, 34,
35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 65, 66, 156, 157, 158, 373, 37
4, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 531, 532, 5
33, 534, 535, 536, 537, 538, 566, 567, 568, 743, 744, 757, 758, 759, 760, 761,
762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777,
778, 779, 780, 1187, 1188, 1189, 1190, 1191, 1229, 1230, 1231, 1232, 1233, 165
2, 1653, 1741, 1770, 1771, 1772, 1773, 1774, 2197, 2198, 2199, 2200, 2201, 220
2, 2203, ...], ('F', '46-50'): [248, 249, 250, 251, 252, 253, 254, 255, 256, 25
7, 414, 415, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 472, 473, 474, 6
54, 655, 656, 657, 658, 717, 718, 719, 720, 721, 722, 723, 724, 725, 879, 880,
881, 895, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1095, 1096, 1097, 1098, 109
9, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 111
2, 1113, 1114, 1115, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 143
4, 1435, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 256
0, 2561, 2562, 2563, 2564, 2565, 2566, ...], ('F', '51-55'): [14, 15, 16, 17, 3
55, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 400,
401, 402, 997, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 147
7, 1478, 1479, 1480, 1957, 1958, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 199
0, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 200
3, 2004, 2005, 2026, 2027, 2028, 2782, 2783, 2784, 3526, 3527, 3528, 3650, 365
1, 3652, 3653, 3654, 3993, 3994, 3995, 3996, 4177, 4178, 4179, 4180, 4755, 490
1, 4902, 4903, 5625, 5626, 5630, 5631, 5632, 5633, 5884, 5885, 5886, 5887, 588
8, 5889, ...], ('F', '55+'): [475, 476, 1961, 1962, 1963, 1964, 1965, 1966, 196
7, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1981, 1982, 213
9, 2140, 2141, 2142, 2227, 2228, 2229, 2230, 2231, 2232, 3123, 3124, 3125, 312
6, 3127, 3128, 3129, 3130, 3131, 3132, 3133, 3134, 3655, 3656, 3657, 3658, 365
9, 3660, 3687, 3688, 3689, 3690, 4693, 5444, 5445, 5446, 5447, 5448, 5449, 545
0, 5451, 5452, 5594, 5595, 5596, 5597, 5732, 5733, 5734, 5735, 5736, 5737, 573
8, 5739, 5740, 5741, 5742, 5743, 5744, 5745, 5765, 5766, 5767, 5768, 5769, 607
7, 6078, 6404, 6405, 6406, 6407, 6408, 6409, 6931, 7780, 7781, 7782, 7783, 822
3, ...], ('M', '0-17'): [85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 865, 8
66, 867, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 224
7, 2248, 2249, 2250, 2251, 3014, 3015, 3016, 3017, 3018, 3568, 3569, 3570, 357
1, 4375, 4526, 4527, 4528, 4529, 4530, 4531, 4532, 4647, 4648, 4649, 4650, 465
1, 4652, 4653, 4654, 4655, 4656, 4698, 4699, 4771, 5059, 5060, 5061, 5062, 506
3, 5064, 5065, 5066, 5352, 5361, 5362, 5435, 5436, 5437, 5438, 5439, 5440, 544
1, 5624, 5725, 5821, 5822, 5823, 5824, 5919, 5920, 5921, 5922, 5923, 5924, 592
5, 6032, 6033, 6112, 6113, 6114, 6115, 6520, 6521, ...], ('M', '18-25'): [97, 9
8, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 127, 128, 12
9, 220, 221, 258, 259, 260, 261, 262, 263, 291, 292, 293, 294, 295, 296, 300, 3
01, 302, 303, 339, 340, 341, 388, 389, 390, 391, 392, 403, 404, 405, 408, 409,
416, 417, 418, 419, 420, 438, 439, 440, 462, 463, 464, 465, 466, 583, 584, 652,
653, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 700, 701, 702, 703, 704,
705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 726, 727, 728, 750,
751, 752, 753, ...], ('M', '26-35'): [5, 9, 10, 11, 12, 13, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28, 50, 51, 56, 57, 58, 59, 60, 61, 62, 63, 64, 130, 131, 132,
133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 196, 197,
198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213,
214, 215, 216, 217, 218, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234,
264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279,
280, 281, 282, 283, 284, 285, ...], ('M', '36-45'): [18, 55, 112, 113, 114, 11
5, 116, 117, 118, 119, 120, 121, 122, 123, 152, 153, 154, 155, 335, 336, 337, 3
38, 393, 394, 395, 396, 397, 398, 421, 422, 433, 434, 435, 436, 437, 491, 492,
493, 494, 544, 545, 546, 551, 552, 553, 554, 555, 556, 557, 580, 581, 605, 606,
607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 623, 624, 626, 627, 628, 629,
630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 665,
666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 830, 831, 832, 833, 834,
...], ('M', '46-50'): [6, 7, 8, 52, 53, 54, 164, 165, 166, 167, 168, 169, 170,
171, 172, 173, 174, 175, 176, 177, 178, 189, 190, 191, 192, 193, 194, 195, 235,
236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 527, 528, 558, 559,
560, 561, 562, 563, 564, 565, 569, 570, 571, 572, 573, 574, 576, 577, 578, 646,
647, 648, 649, 650, 651, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908,
909, 1057, 1058, 1089, 1090, 1091, 1307, 1308, 1309, 1323, 1324, 1325, 1326, 13
27, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, ...], ('M', '51-55'):
[67, 68, 69, 333, 334, 370, 371, 788, 789, 790, 791, 792, 793, 794, 795, 796, 7
97, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 868, 869, 870, 871, 1047,
1048, 1049, 1050, 1486, 1487, 1488, 1489, 1503, 1504, 1505, 1506, 1681, 1682, 1
683, 1738, 1739, 1740, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1782, 1815, 18
16, 1817, 1818, 1819, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 2029, 203
0, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 204
3, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 205
6, 2124, 2175, ...], ('M', '55+'): [4, 159, 160, 161, 162, 451, 452, 453, 477,
478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 645, 659, 893,
894, 1051, 1052, 1053, 1054, 1116, 1117, 1118, 1119, 1559, 1560, 1792, 1793, 17
94, 1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 197
8, 1979, 1980, 2016, 2017, 2018, 2096, 2097, 2322, 2510, 2614, 2615, 2766, 276
7, 2768, 2769, 2770, 2771, 2793, 2794, 2795, 2796, 2797, 2798, 2799, 3011, 347
8, 3837, 3838, 3839, 3840, 3841, 3842, 3843, 3844, 3845, 4175, 4176, 4423, 442
4, 4425, 4426, 4694, 4695, 4696, 5052, 5053, 5083, 5084, ...]}
```

In [ ]:

```
```